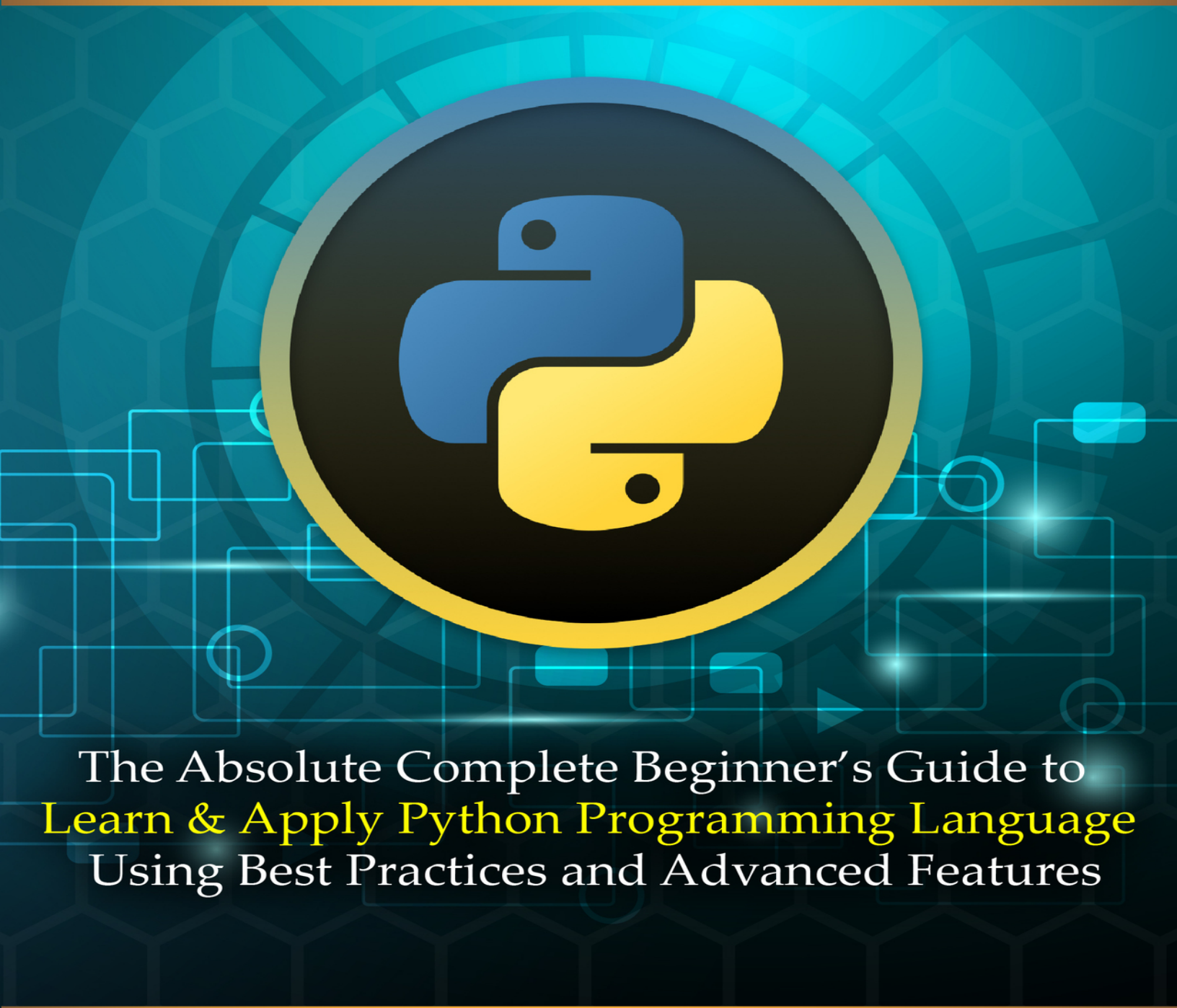
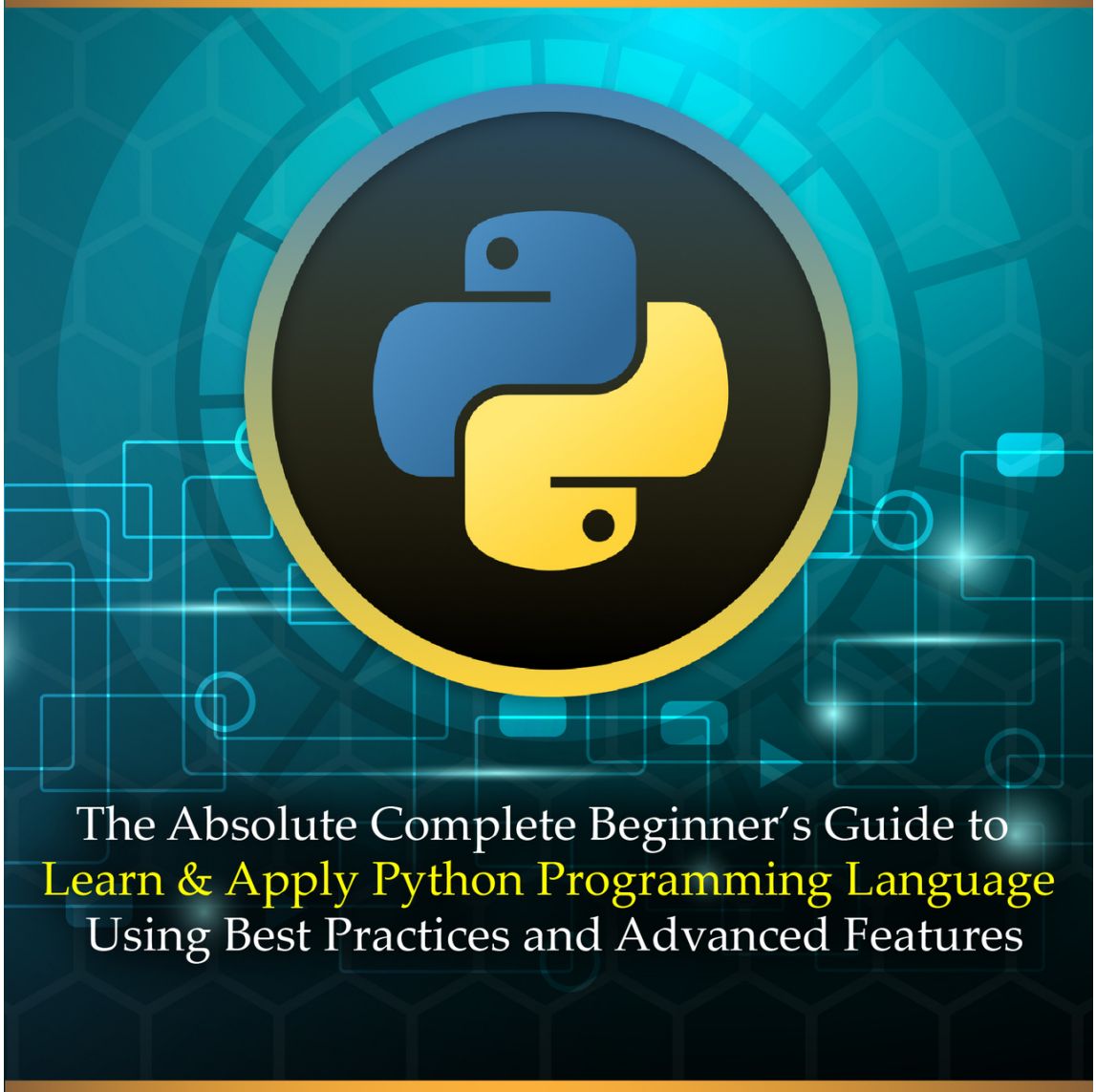
PYTHON PROGRAMMING for BEGINNERS



The Absolute Complete Beginner's Guide to
Learn & Apply Python Programming Language
Using Best Practices and Advanced Features

JAMES HERRON

PYTHON PROGRAMMING for BEGINNERS



The Absolute Complete Beginner's Guide to
Learn & Apply Python Programming Language
Using Best Practices and Advanced Features

JAMES HERRON

Python Programming For Beginners

The Absolute Complete Beginner's Guide to Learn and Apply Python Programming Language Using Best Practices and Advanced Features.

James Herron

Table Of Contents

Intro to Python

- What is Python?
- Features of Python
- Huge set of libraries
- What sort of application I can create using Python?
- Who uses Python?

Chapter 1 Installing Python

- Choosing a Python Version
- General installations instruction
- Installation on Windows
- Installation on Linux (Ubuntu)
- Installation on Mac OS
- Running Programs
- Interactive Interpreter or Interactive Mode via Shell
- Script from instruction
- Python IDE (Integrated Development Environment)

Chapter 2 IDLE and Python Shell

- Installing the Interpreter
- Python IDLE

Chapter 3 Python Data Types

- Python Numbers
- Python Strings

Chapter 4 Python Variables

- What Is A Variable in Python?
- Classifications of Python Arrays Essential for Variables
- Learning Python Strings, Numbers and Tuple
- Types of Data Variables
- Declaring Variables

Variables Assignment
Functions and Sets In Python Variables
Chapter 5 Data Structures
Control Flow Statements
Nested If
While Loop
For Statement
Functions
Rules to outline a operate in Python
Control Statements
String Manipulation
Exception Handling
Chapter 6 Learning concerning Functions
Understanding the conception of operate
Using numerous Functions
Chapter 7 Conditional and Loops in Python
What is a sequence in Python?
Selection method management
Chapter 8 Inheritance and Polymorphism
Creating a category in Python
Class Attributes
Class Data Attributes
Python Class Inheritance
Why is Inheritance Useful in Python Programming?
Inheritance Example
Class Polymorphism and Abstraction
Abstraction
Encapsulation
Conclusion

Intro to Python

This book is about Python for beginners. It introduces the core aspects of the Python programing language . Python may be a high-level, integrated, general-purpose programing language developed in 1991 by Guido van Rossum. the planning philosophy underpinning Python emphasizes on code readability characterized by its use of considerable whitespace. Python's object-oriented approach and language constructs specialize in helping programmers write logical, clear code for giant and small-scale projects. This book aims to supply a cogent introduction to python for beginners. It seeks to supply a platform to find out python programming well and in one week including step by step practical examples, exercises and tricks.

Before we enter the' why programming' discussion first allow us to knows what programming is. Programming is that the process of taking an equation and converting it into a code, a programing language , in order that it's implemented on a machine. Or in simple words, "Programming may be a language to show a machine what to try to through a group of instructions." Various sorts of programming languages are used, for example:

- Python
- PHP
- C language
- JAVA and more.

So why is it important? What's so Great about this? Why does it matter?

The first thing, everything is completed on computers in today's world; from sending an email / report back to a foreign colleague / friend / relative or maybe as simple as an image to having a crucial meeting on Skype! it's become a necessity for each single individual to possess a computer / laptop because it is fast, very reliable and straightforward to use. So when computers are a part of your life, it's said that learning to program will boost your life! one among the most reason people are learning programming is because they need

to form a career by creating websites for companies or mobile apps. that's not the sole reason you would like to find out programming; programming also can help to enhance an individual's efficiency and productivity!

What is Python?

Python may be a multi-purpose language created by Guido van Rossum. Differently from HTML, CSS, and JavaScript, Python is often used for multiple sorts of programming and software development. Python are often used for things like:

- Back end software development
- Desktop apps
- Big data and mathematical computations
- System scripts
- Data Analysis
- Data Science
- Artificial Intelligence
- and lots of others

The reasons why Python is that the attend language for several are often summarized within the following points:

- Beginner Friendliness because it reads like almost English
- Very no hard rules on the way to build features
- Easier to manager error
- Many and large supporting communities
- Careers opportunities
- Future primarily for the mighty fit towards Data Science and Machine Learning Applications.

Python is one among the foremost powerful programming/coding languages. It’s one among the programming languages that are interpreted instead of compiled. this suggests the Python Interpreter works or operates on Python programs to offer the user the results. The Python Interpreter works during a line-by-line manner. With Python, one can do tons . Python has been used for the event of apps that span a good of fields, from the foremost basic apps to the foremost complex ones. Python are often used for development of

the essential personal computer applications. it's also an honest coding language for web development. Websites developed with Python are known for the extent of security and protection they supply , making them safe and secure from hackers and other malicious users. Python is well applicable within the field of game development. it's been used for the event of basic and sophisticated computer games. Python is currently the simplest programing language to be used in data science and machine learning. it's libraries that are best suitable to be used in data analysis, making it suitable to be used during this field. an honest example of such a library is scikit-learn (sklearn) which has proved to be the simplest to be used in data science and machine learning.

Python is documented for its easy-to-use syntax. it had been written with the goal of creating coding easy. This has made it easy the simplest language even for beginners. Its semantics also are easy, making it easy for one to know Python codes. The language has received tons of changes and enhancements , especially after the introduction of Python 3. Before, we had Python 2.7 which had expanded much stability. Python 3 brought in new libraries, functions, and other features, and a few of the languages construct where changed significantly.

Features of Python

Python is straightforward

Python is a simple language to urge started with. the convenience of use is underscored by the very fact that the majority of the programs written in python look almost like English language. Therefore, this simplicity makes python a perfect learning language for entry-level programming courses, and thus introducing programming concepts to students.

Python is Portable/Platform Independent

Python is massively portable, which suggests that Python programs are often run in various different operating systems without needed specific or extensive changes.

Python is an Interpreted Language

Mainly, Python is an interpreted language as against being a compiled language on the opposite hand, C, C++ are samples of complied languages.

In many cases, the programs composed during a application-oriented language are typically mentioned ASCII text file or source programs. As a result, the commands within the source codes are mentioned as statements. A computer lacks the capacity to execute a program written in application-oriented language . Typically, computers know machine language, which contains 1s and 0s only.

As a result, there are two sorts of program available to users when translating high-level languages to machine languages: compiler and interpreter.

Compiler

In its operative function, a complier translates the whole ASCII text file into the readable machine language in one swoop. The machine language is subsequently executed.

Interpreter

On the opposite hand, an interpreter employs a line by line approach to translate a application-oriented language into machine language, which is subsequently executed. The Python interpreter bags are the highest of the file, translates the start line into machine language, and subsequently executes it.

It is crucial to differentiate between high-level and complied languages. for instance , the compiled languages like C, and C++ use a compiler to translate and interpret (high-level to machine language.) On the opposite hand, an interpreted language like Python employs an interpreter to conduct this approach of translation and subsequent execution. The important distinction here between interpreted and compiled language is that the widely complied language operates and performs better compared to written programs that employ interpreted languages. However Python doesn't suffer from this disadvantage.

In synthesis, Python is an interpreted language because the program executes directly from the ASCII text file . whenever Python programs are run the ASCII text file is required. What Python does is to convert the ASCII text file written by the developer into an intermediate code which is then further converted into the machine language able to be executed. Python is therefore an Interpreted language because it is processed in real time by the interpreter. The ASCII text file doesn't got to be compiled before its execution. Compiled means the code must be converted to machine language before runtime.

Python is Dynamically Typed

Another characteristic of Python is to be dynamically typed which suggests that data types are checked on the fly, during execution vs statically typed when data type are checked before run-time.

Python is Strongly Typed

Python is Strongly Typed

The primary feature of a strongly typed language is that it lacks the capacity to convert one form (type) of knowledge to a different automatically. On the opposite hand, languages like PHP and JavaScript, which are loosely typed, have the capacity to convert data from one type to a different freely and automatically.

In this regard, before adding 12 to the strong, the JavaScript language seeks to convert the amount 12 to a robust “12”, which is subsequently appended to the top of the strong.

The language (Python) would occasion a mistake because it doesn't have the capacity to convert the umber 12 into a robust .

In synthesis Python may be a strongly typed language i.e. the sort of variables must be known. this suggests that the sort of a worth doesn't change in unexpected ways. for instance a string containing only number doesn't automatically become variety , as may happen in Perl. In Python, every data type change needs a particular conversion.

Huge set of libraries

Python provides users with a broad range of libraries, which make it easy to feature new capacities and capabilities without necessarily reinventing new approaches.

What sort of application I can create using Python?

Some of the most applications that Python is employed include:

Games

- Machine Learning and AI
- Data Science and Data Visualization
- Web Development
- Game Development
- Desktop GUI
- Web Scraping Applications
- Business Applications
- Audio and Video Applications
- CAD Applications
- Embedded Applications
- System administration applications
- GUI applications.
- Console applications
- Scientific applications
- Android applications

And several others not mentioned here.

Who uses Python?

Some of the most companies that utilize Python include:

I. YouTube
II. Mozilla
III. Dropbox
IV. Quora
V. Disqus
VI. Reddit
VII. Google
VIII. Disney
IX. Mozilla

X. Bit Torrent
XI. Intel
XII. Cisco

xiii. Banks like JPMorgan Chase, UBS, Getco, and Citadel apply Python for financial market forecasting.

xiv. NASA for scientific programming tasks

xv. iRobot for commercial robotic vacuum

Chapter 1

Installing Python

To code in Python, you want to have the Python Interpreter installed in your computer. you want to even have a text editor during which you'll be writing and saving your Python codes. the great thing with Python is that it can run on various platforms like Windows, Linux, and Mac OS. Most of the present versions of those operating systems come installed with Python. you'll check whether Python has been installed on your OS by running this command on the terminal or OS console:

Python

Type the above command on the terminal of your OS then hit the Enter/Return key.

The command should return the version of Python installed on your system. If Python isn't installed, you'll be told that the command isn't recognized; hence you've got to put in Python.

Choosing a Python Version

The central two reports of Python are 2.x and 3.x. Python 3.x is clearly the newest one but Python 2.x as of today is presumably still the foremost used one. Python 3.x is however growing much faster in terms of adoption. Python 2.x remains in use in many software companies. More and more projects still are running to Python 3.x. There are several technical differences between the two versions. we will summarize in very an easy way as Python 2.x is legacy and Python 3.x is that the future. the recommendation for you is to travel for the newest version Python 3.x. From 2020 Python 2.x isn't be supported anymore.

General installations instruction

Installing Python is extremely easy. All you would like to try to do is follow the steps described below:

1) Attend Python downloads page https://www.python.org/downloads/

2) Click the link associated with your OS

3) Click on the newest release and download consistent with your OS

4) Launch the package and follow the installation instructions (we recommend to go away the default settings)

Make sure you click on Add Python 3.x to PATH. Once the installation is finished you're set to go!

5) Access your terminal IDLE

Test that each one works by writing your first Python code:

☐ print ("I'm running my first Python code")

press enter or return, this is often what you ought to get

You can do an equivalent also by launching this command employing a file. we'll address this after we address the Python IDLE or another code editor.

Installation on Windows

To install Python on Windows, download Python from its official website then double click the downloaded setup package to launch the installation. you'll download the package by clicking this link:

https://www.python.org/downloads/windows/

It will be good for you to download and install the newest package of Python as you'll be ready to enjoy using the newest Python packages. After downloading the package, double click thereon and you'll be guided through on-screen instructions on the way to install Python on your Windows OS.

Installation on Linux (Ubuntu)

In Linux, there are variety of package managers which will be used for installation of Python in various Linux distributions. for instance ,

if you're using Ubuntu Linux, run this command to put in Python:

$ sudo apt-get install python3-minimal

Python are going to be installed on your system. However, most of the newest versions of varied Linux distributions come installed with Python. Just run the “python” command. If you get a Python version because the return, then Python has been installed on your system. If not, plow ahead and install Python.

Installation on Mac OS

To install Python in Mac OS, you want to first download the package. you'll find it by opening the subsequent link on your web browser:

https://www.python.org/downloads/mac-osx/

After the setup has been downloaded, double click it to launch the installation. you'll be presented with on screen instructions which will guide through the installation process. Lastly, you'll have Python running on your Mac OS system.

Running Programs

One can run Python programs in two main ways:

- Interactive interpreter
- Script from instruction

Interactive Interpreter or Interactive Mode via Shell

Python comes with a instruction which is usually mentioned because the interactive interpreter. you'll write your Python code directly on this interpreter and press the enter key. you'll get instant results. If you're on Linux, you simply need to open the Linux terminal then type “python”. Hit the enter key and you'll be presented with the Python interpreter with the >>> symbol. To access the interactive Python interpreter on Windows, click Start -> All programs, then identify “Python …” from the list of programs. In my sample, I obtain“Python 3.5” as I also have installed Python 3.5. Expand this feature and click on “Python …”. In my case, I click “Python 3.5(64-bit)” and that i get the interactive Python interpreter.

Here, you'll write and run your Python scripts directly. to write down the “Hello” example, type the subsequent on the interpreter terminal:

print("Hello")

Hit the enter/return key and therefore the text “Hello” are going to be printed on the interpreter:

Script from instruction

This method involves writing Python programs during a file, then invoking the Python interpreter to figure on the file. Files with Python should be saved with a .py extension. this is often a designation to suggest that it's a Python file. for instance , script.py, myscript.py, etc. After writing your code within the file and saving with the name “mycode.py”, you'll open the OS instruction and invoke the Python interpreter to figure on the file. for instance , you'll run this command on the instruction to execute the code on the file mycode.py:

☐ python mycode.py

The Python guide will run on the file and print the results on the terminal.

Python IDE (Integrated Development Environment)

If you've got a GUI (Graphical User Interface) application capable of supporting Python, you'll run the Python on a GUI environment. the subsequent are the Python IDEs for the varied operating systems:

- UNIX- IDLE
- Windows- PythonWin

Macintosh comes along side IDLE IDE, downloadable from the official website as MacBinary or BinHex'd files.

Chapter 2

IDLE and Python Shell

As we've seen Python are often utilized in 2 main modes:

1) Interactive mode also referred to as via Python Shell
2) Via Python IDLE

As a reminder the Python Shell referred to as the prompt string, is prepared to simply accept commands. The Python shell allows typing Python code and getting the result immediately. The Python Shell is sweet for testing small chunk of code. The Python IDLE (IDLE or IDE stands for Integrated Development Environment) instead allows to try to that also but gives far more functionalities. Therefore we advise you to travel straight for the Python IDLE. so as to start out using Python IDLE we recommend also to make a replacement directory for instance “PythonPractice” where you would like , in there you'll save future python files.

Installing the Interpreter

As a reminder if not done yet before we will write our first Python program, we've to download the acceptable interpreter for our computers.

We’ll be using Python 3 during this book because as stated on the official Python site “Python 2.x is legacy; Python 3.x is that the present and way forward for the language”. “Python 3 excludes many quirks which will accidentally trip up origin programmers”. Yet, regard that Python 2 is currently yet rather generally used. Python 2 and three are about 90% similar. Hence if you learn Python 3, you'll likely haven't any problems understanding codes written in Python 2.

To install the guide for Python 3, origin over to

https://www.python.org/downloads/ .

The correct version should be indicated at the highest of the webpage. We’ll be using version 3.6.1 during this book. Click on

“Download Python 3.6.1” and therefore the software will start downloading.

Alternatively if you would like to put in a special version, scroll down the page and you’ll see an inventory of other versions. Click on the discharge version that you simply want. You’ll be redirected to the download page for that version.

Scroll down towards the top of the page and you’ll see a table listing various installers for that version. Choose the right installer for your computer. The installer to use depends on two factors:

1. The OS (Windows, Mac OS, or Linux) and
2. The processor (32-bit vs 64-bit) that you simply are using.

For instance, if you're employing a 64-bit Windows computer, you'll likely be using the "Windows x86-64 executable installer". Just click on the link to download it. If you download and run the incorrect installer, no worries. you'll get a mistake message and therefore the interpreter won't install. Simply download the right installer and you're good to travel . Once you've got successfully installed the interpreter, you're able to start coding in Python.

Python IDLE

We advise to use a minimum of the default Python IDLE. However there are many other options you'll use the table below to match them. IDLE is that the integrated development environment for Python and it's installed automatically with Python. also as a neat graphical interface , IDLE is full of features that make using Python for developing easy, including a really powerful feature, syntax highlighting.

With syntax highlighting, reserved keywords, literal text, comments, etc. are all highlighted in several colors, making it much easier to ascertain errors in your code. also as editing your Python program with IDLE, you'll also execute your programs in IDLE.

Chapter 3

Python Data Types

Python supports different data types. Each variable should belong to at least one of the info types supported in Python. the info type determines the worth which will be assigned to a variable, the sort of operation which will be applied to the variable also because the amount of space assigned to the variable. Let's discuss different data types supported in Python.

Python Numbers

These data types help within the storage of numeric values. The creation of number-objects in Python is completed after we've assigned a worth to them. Consider the instance given below:

total = 55

age = 26

The statement are often used for the deletion of single or multiple variables. this is often shown below:

del total

del total, age

In the first statement, we are deleting one variable while within the second statement, we are deleting two variables. If the variables to be deleted are quite two, separate them by employing a comma and that they are going to be deleted.

In Python, there are four numerical values which are supported:

- Int
- Float
- Complex

In Python 3, all integers are represented within the sort of long integers.

The Python integral literals regard to the int class.

Example

Run the subsequent statements consecutively on the Python interactive interpreter:

```
x=10
x
```

You can run it on the Python interactive interpreter and you'll observe the following:

The float is employed for storing numeric values with a percentage point .

Example

```
x=10.345
x
```

If you're performing an operation with one among the operands being a float and therefore the other being an integer, the result are going to be a float.

Example

```
5 * 1.5
```

As shown above, the results of the operation are 7.5, which may be a float.

Complex numbers are made from real and imaginary parts, with the imaginary part of a complex number being denoted employing a j. they will be defined as follows:

```
x = 4 + 5j
```

4 is that the real part, while 5 is that the imaginary part of a complex number.

In Python, there's a function named type() which will be used for determining the sort of a variable. you simply need to pass the name of the variable inside that function because the argument and its type are going to be printed.

Example

```
x=10

type(x)
```

The variable x is of int class as shown above. you'll try it for other variable types as shown below:

```
name='nicholas'

type(name)
```

Python Strings

Python strings are a set of parts contained within quotes. Use any sort of quotes to surround Python strings, that is, either single, double or triple quotes. To enter string details, we apply the slice operator. String characters begin at index 0, meaning that the primary character string is at index 0. this is often good once you got to access string characters. To concatenate strings in Python, we use + operator, the asterisk 9*) is employed for repetition.

Example

```
#!usrbin/Python3

thanks = 'Thank You'

print (thanks) # to print the entire string

print (thanks[0]) # to print the primary character of the string

print (thanks[2:7]) # to print the 3rd to the 7th character of the string

print (thanks[4:]) # to print from the 5th character of the string

print (thanks * 2) # to print the string twice

print (thanks + "\tAgain!") # to print a concatenated string
```

The program prints the subsequent once executed:

Notice that we've text beginning with the # symbol. The symbol denotes the start of a comment. The Python print won't act on the text from the symbol to the top of the road . Judgments are intended at enhancing the readability of code by providing information. We

defined a string named thanks with the worth many thanks . The print (thanks[0]) statement helps us access the primary character of the string; hence it prints T. you furthermore may notice that the space between the 2 words is counted as a personality .

Chapter 4

Python Variables

Understanding Python variables, classes, and the way to work is important for both beginners and programmers who shall expand their programming skills.

What Is A Variable in Python?

When writing complex codes, your program will demand data essential to conduct changes once you proceed together with your executions. Variables are, therefore, sections wont to store code values created after you assign a worth during program development. Python, unlike other related language programming software, lacks the command to declare a variable as they modify after being set. Besides, Python conditions are limitless like in most cases of programming in different computer languages.

Variation in Python is hence defined as memory resources used for saving data values. As such, Python variables act as storage units, which feed the pc with the required data for processing. Each value comprises of its database in Python programming, and each data are categorized as Numbers, Tuple, Dictionary and List, among others. As a programmer, you understand how variables work and the way helpful they're in creating an efficient program using Python. As such, the tutorial will enable learners to know declare, re-declare, and concatenate, local and global variables also as the way to delete a variable.

Variable vs. Constants

Variables and constants are components utilized in Python programming but perform different functions. Variables, also as constants, utilize values wont to create codes to execute during program creation. Variables act as essential storage locations for data within the memory, while constants are variables whose value remains unchanged. as compared , variables store reserves for data

while constants are a kind of variable files with consistent values written in capital letters and separated by underscores.

Variables vs. Literals

Variables are also a part of literals which are data ate up either variable or constant with several literals utilized in Python programming. a number of the common sorts of literals used include Numeric, String, and Boolean, Special and Literal collections like Tuple, Dict, List, and Set. The difference between variables and literals arises where both affect unprocessed data but variables store the while laterals feeds the info to both constants and variables.

Variables vs. Arrays

Python variables have a singular feature where they only name the values and store them within the memory for quick retrieval and supplying the values when needed. On the opposite hand, Python arrays or collections are data types utilized in programing language and categorized into list, tuple, set, and dictionary, which can be discussed later. in comparison to variables, the array tends to supply a platform to incorporate collectives functions when written while variables store all types of knowledge intended. When choosing your charming collection, make sure you select the one that matches your requirements henceforth meaning retention of meaning, enhancing data security and efficiency.

Classifications of Python Arrays Essential for Variables

Lists

Python lists offer changeable and ordered data and written while accompanying square brackets, for instance, "an apple," "cherry." Accessing an already existing list by pertaining to the index while with the power to write down negative indexes like '-1' or '-2'. you'll also maneuver within your list and choose a selected category of indexes by first determining your starting and endpoints. The return charge with hence be the range of defined items. you'll also specify a scale of negative indexes, alter the worth of the present item, the

loop between items on the list, add or remove items, and confirming if items are available.

Dictionaries

Python dictionaries comprise of indexed, changeable but unordered items typically written while with curly brackets with keys and values. a number of the activities involved include item access by use of a keyword inside the parentheses; conduct value changes, loop, check critical availability, length of the dictionary, and both adding and removing unwanted items. Besides, Python allows you to repeat the dictionary by writing 'dict2 = dict1'. 'dict2' will match a design to 'dict1' hence makes any required revisions automatically. differently of making a replica is additionally by employing a built-in Dictionary technique, that is, 'copy.'

In other instances, Python dictionaries also can produce other dictionaries within it a process mentioned as nested dictionaries. you'll readily determine the amount of dictionaries present within the nest through creating of three already available. you'll also generate your dictionary through the 'dict()' contractor function. The function enables the copying of the previous dictionary or the creation of a totally new one. Within the Python dictionary, there exist several built-in techniques to implement and luxuriate in the efficiency of the dictionaries present.

Naming Variables

The naming of variables remains straightforward, and both beginners and experienced programmers can readily perform the method . However, providing titles to those variables accompany specific rules to make sure the supply of the proper name. Consistency, style, and adhering to variable naming rules make sure that you create a superb and reliable name to use both today and therefore the future. the principles are:

- Names must have one word, that is, with no spaces
- Names must only comprise of letters and numbers also as underscores like (_)
- The first letter must not ever be variety

- Reserved words must not ever be used as variable names

When naming variables, you ought to bear in mind that the system is case-sensitive, hence avoid creating equivalent names within one program to stop confusion. Another important component when naming is considering the design. It involves beginning the sign with a lowercase letter while applying underscores as terms between your words or phrases applied. Besides, the program customarily prevents starting the name with a capital . Begin with a lowercase letter and either mix or use them consistently.

When creating variable names, it's going to seem very easy , but sometimes it's going to become verbose henceforth becoming a disaster to beginners. However, the challenge of making sophisticated names is sort of beneficial for learned because it prepares you for the subsequent tutorials. Similarly, Python enables you to write down your required name of any length consisting of lower- and upper-case letters, numbers also as underscores. Python also offers the addition of complete Unicode support essential for Unicode features in variables.

As already discussed, specific rules are governing the procedure for naming variables; hence adhere to them to make an exceptional name to your variables. Create more readable names that have aiming to prevent instances of confusion to your members, especially programmers. A more descriptive name is far preferred compares to others. However, the technique of naming variables remains illegible as different programmers choose how they're getting to create their quite names.

Methods of making a Multi-Name for Python Variables

- Pascal case: this method entails the primary , second, and subsequent words within the name as capitalized to reinforce readability. for instance , Concentration Of White Smoke.

- Camel case: the second and subsequent words of the name created remains capitalized. for instance , the Concentration of White Smoke.

- Snake case: snake method of making variable names entails separator of words using an underscore as mentioned earlier. for instance , concentration_of_white_smoke.

Learning Python Strings, Numbers and Tuple

Python strings are a part of Python variables and comprise of objects created from enclosing characters or values in double-quotes. as an example , 'var = Hello World'. With Python not supporting character types in its functions, they're however treated as strings of 1 more characters also as substrings. Within the Python program, there exist several string operators making it essential for variables to be named and stored in several formats. a number of the string operators commonly utilized in Python are [], [:], 'in', r/R, %, + and *.

There exist several methods of strings today. Some include replacing Python string () to return a replica of the previous value during a variable, changing the string format, that is, upper and lower cases and using the 'join' function, especially for concatenating variables. Other programs involve the reverse function and cut strings applying the command' word. split'. What to notice is that strings play a crucial role, especially in naming and storage of values despite Python strings being immutable.

On the opposite hand, Python numbers are categorized into three main types; that's , int, float, and sophisticated . Variable of numbers are usually created when assigning a worth for them. as an example , int values are generally whole numbers with unlimited length and are either positive or negative like 1, 2, and 3. Float numbers also either positive or negative and should have one or more decimals like 2.1, 4.3 and 1.1 while complex numbers comprise both of a letter 'j' because the imaginary portion and numbers, for instance , 1j, -7j or 6j+5. on verify the variable number may be a string, you'll readily use the function 'type().'

A collection of ordered values, which remain unchangeable, especially in Python variables, is mentioned as a tuple. Python tuples are indicated with round brackets and available in several ways. Some useful in Python variables are access tuple items by index numbers and inside square brackets. Added is tuple continuing

stable, particularly after being built but renders a loop by applying the function 'for.' And it readily encompasses both count and index methods of tuple operations.

Types of Data Variables

String

A text string may be a sort of data variable represented in either String data types or creating a string from a variety of type char. The syntax for string data comprises multiple declarations including 'char Str1[15], 'char Str5[8] = “ardiono”; among others. on declare a string effectively, add null character 'Str3', declare arrays of chars without utilizing within the sort of 'Str1' and initialize a given array and leave space for a bigger string like Str6. Strings are usually displayed with doubles quotes despite the several versions of obtainable to construct strings for varying data types.

Char

Char are data types primarily utilized in variables to store character values with literal values written in single quotes, unlike strings. The values are stores in numbers form, but the precise encoding remains visibly suitable for performing arithmetic. as an example , you'll see that it's saved as 'A' +, but it's a worth of 66 because the ASCII 'A' value represents 66. Char data types are usually 8 bits, essential for character storage. Characters with larger volumes are stored in bytes. The syntax for this sort of variable is 'char var = val'; where 'var' indicates variable name whileval’ represents the worth assigned to the variable.

Byte

A byte may be a data type necessary for storing 8-bit unsigned numbers that are between 0 to 255 and with a syntax of 'byte var = val;.' Like Char data type, 'var' represents variable name whileval’ stands for the worth to he assigned that variable. The difference between char and byte is that char stores smaller characters and with a coffee space volume while byte stores values which are larger.

int

Another sort of data type variable is that the int, which stores 16-bit value yielding an array of between -32,768 and 32,767, which varies counting on the various programming platforms. Besides, int stores 2's complement math, which is negative numbers, henceforth providing the potential for the variable to store a good range of values in one reserve. With Python, this sort of knowledge variable storage enables transparency in arithmetic operations in an intended manner.

Unsigned int

Unsigned int also mentioned , as unsigned integers are data types for storing up to 2 bytes of values but don't include negative numbers. The numbers are all positive with a variety of 0 to 65,535 with Duo stores of up to 4 bytes for 32-byte values, which range from 0 to 4,294,967,195. as compared , unsigned integers comprise positive values and have a way higher bit. However, ints take mostly negative values and have a lower bit hence store chapters with fewer values. The syntax for unsigned int is 'unsigned int var = val;' while an example code being 'unsigned int ledPin = 13;'

Float

Float data types are values with point numbers, that's to mention , variety with a percentage point . Floating numbers usually indicate or estimate analog or continuous numbers, as they possess a more advanced resolution compared to integers. The numbers stored may range from the very best of seven .5162306E+38 and therefore the lowest of -3.2095174E+38. Floating-point numbers remain stored within the sort of 32 bits taking about 4 bytes per information fed.

Unsigned Long

This is data sorts of variables with an extended size; hence it stores values with larger storages compare to other data types. It stores up to 32 bits for 4 bytes and doesn't include negative numbers henceforth features a range of 0 to 4,294,967,295. The syntax for the unsigned long data type is 'unsigned long var = val;' essential for storing characters with much larger sizes.

Declaring Variables

As stated, variables are the naming and storing data values of both numerical and letters primarily used during Python programming. Before those values are used, they need to be declared to spot and perform the specified function in your program. Therefore, declaring a worth means defining the sort , and sometimes setting or initializing the worth , though it remains optional but crucial. When creating your program, make sure you understand the extent of your variables by considering the scope of numbers you're storing. Excessive storage of values may end in rollovers, because the space used is insufficient.

The size of where to declare your values also affects a programmer's outcome on created programs. The technique of choosing a selected storage unit influences the function of applications, especially when determining the codes. The scope henceforth is an important aspect of declaring variables because it affects the results of your program. Another form is thru initializing variables to make a decision on which value to start out with during declaration. Initialized variables make it easy for programmers to readily choose a start line when declaring or used for other purposes.

Creating and Declaring Variable

Python programs have limited access to an immediate command to declare or create variables instantly. However, some essential rules may become a critical component for the method to occur. Besides, Python doesn't necessarily require data type specification but created immediately when the worth is assigned. When assigning values using Python, especially for complex or multiple assignments, it uses inferred language techniques, for instance, in detecting sorts of values assigned during a given variable.

Variables Assignment

Variables are neither declared nor defined when utilizing Python programming henceforth creation is sort of straightforward. Creating a variable is just assigning a worth and begins using it. the method uses one equal (=) symbol useful for statements and expressions. for instance , creating n=38 in Python suggests that 'n' is assigned

the worth '38' and therefore the value are often readily be substituted during programming.

Like literals values, the worth used could also be displayed directly by the interpreter without the utilization of 'print().' However, if you modify the worth , rather than 'n=38', the worth are going to be substituted; as an example , if you input value 1000, it'll display 'n=1000'. Python henceforth allows you to form changes where needed also because the operation of chained assignments and input an identical value to different variables simultaneously. for instance , a=b=c=d=38.

Re-Declaring Variables

After you've got even declared a variable in Python, you'll make changes by either declaring it again or assign a substitute value. That is, you'll replace or connect a special value to the previous one readily through the re-declaration process. Re-declaring variables are beneficial because it enables you to simply accept user-generated codes to already existing values initialized. Similarly, you'll wish to vary the program or make some alterations during your project.

Reassigning variables are more vital for sophisticated or extensive programs already incorporated by another programmer, and you're taking up . As such, you create significant changes within the declared values to reinforce the effectiveness of your program is made . The Python interpreter plays a considerable role in discarding the first value and adding the new ones. the sort of latest values attached may either vary or comprise of unique identity in comparison to the old ones.

For example, if your original value was 'x' and you would like to vary it to be an integer of '76', you initially reassign 'x’ to be a string, ass your new value and a replacement of the worth immediately becomes a hit . the instance suggests that value 'x' undergoes an assignment to the worth of an integer and later reassigned with the worth of the string. Variable re-declare is best once you are conscious of the readability of codes and with an object to make clear programs.

Concatenate Variables

If you would like to concatenate variables of several data types through Python, as an example , several variables and string variables, then you would like to declare the values into strings. just in case the amount variables aren't declared when concatenating different values, Python programming would stop and display TypeError indicating the procedure is unsuccessful. on correct things , you'll need to declare the amount variables as a string.

The process at Python programming is sort of different in comparison to other programs like Java and C++, which immediately concatenates several numbers without declaring into strings. as an example , if you would like to concatenate 'computer and '58', however once you declare the integer as a string, it can readily concatenate the 2 within the sort of “computer + str(58) = computer58”.

Global Variables

Global variables utilized in Python are a multipurpose variable utilized in any a part of the planet while anywhere. The variable used can operate in your program or module while in any a part of the world henceforth using values whenever you travel as a programmer. Global variables are useful for programmers to make their programs while moving from one location to a different . a number of the advantages include variables that are used across function or module also because it doesn't require re-declarations for performance.

When related to local variables, global variables have an 'f' scope and detailed value 101, presented as 'f=101', issued as an output. for instance, once you re-declare a variable as a worldwide variable during a given function, change it within the role and print it outside the task. The variable would offer a 3rd party outcome useful globally. Therefore, global variables are found outside functions indicating that not all variables are readily accessed from anywhere globally. As a beginner, it's crucial to know the difference between global and native variables to develop the required variables suitable for your programs.

Local Variables

Unlike global variables, local variables are used locally, declared within a Python function or module, and utilized solely during a specific program or Python module. When implemented outside particular modules or tasks, the Python interpreter will fail to acknowledge the units henceforth throwing a mistake message for undeclared values. Like global variables, local variables use the 'f' variable where it's declared to assume local scope and assigned 'I am learning Python' then recognized as an area variable.

For example, once you declare the variable ‘f’ the second time, it changes to a replacement function and leads to an area variable. As such, once you accept that variable within the inside function, the method will run with none problems. The execution is formed possible because the second print(f) produces a worth assigned to 'f' as Intellipaat. Whereas, once you print the worth outside the function 'f', it leads to a worth assigned thereto , which is outside the function, that is, a 3rd print(f). therein case, local variables are used only in two surrounding environments of Python programming with those outside the function resulting in failure of operations unless declared.

Using Variables

Immediately the variables are declared, they will be readily defined by setting those adequate to the worth you plan to store with the only same sign mentioned because the assignment operator. the only equal sign, therefore, enables the program to place the specified variable either on the proper or left side on whichever side. After seeding the variables, that is, assigned each variable with a worth , test-specific values to work out if suits the program or use it directly.

For instance, you'll use certain codes to check if the inputVariable2 is a smaller amount than 50, thus set a delay time supported it with a minimum of fifty . the instance therefore tests the variable ‘if (inputVariable2 <50)’ and sets a given variable if it's in ‘inputVariable2 = 50’ and delays the results mistreatment the operate ‘delay(inputVariable2)’. once mistreatment variables, guarantee they need a a lot of descriptive name for readability functions. It conjointly

allows you, and some other person understands what the variable entails further as in recognition throughout the programming method.

Deleting Variables

On the opposite hand, if you create an error, get obviate your current project or simply doing away with everything, Python provides a helpful feature to delete any variable and build extra space for storage of different values. Similarly, there are some unwanted variables, and you want to induce obviate, the delete variable is henceforward the selection for you. once deleting a variable, you must use caution on avoid deleting essential values inside a variable. As such, make sure you apprehend the name of the variable to delete.

If you're deleting a file with a variable name, that doesn't match, the Python interpreter usually throws a mistake message;' NameError: name (filename) isn't known.' on delete a variable effectively, there exists a command inside the program; del' variable name' and also the variable can take away instantly. make sure you input the proper name before running the command. on ensure if your file is deleted, you'll try and print, and if you see a mistake message, it indicates that the variable was deleted with success.

Functions and Sets In Python Variables

There are borderline possibilities of a beginner to grasp variables and also the general operation of making Python programs before understanding what are sets and functions as utilized in variables. once mentioning sets, objects, and functions, it suggests that a special factor once it involves computer programing. one in all the foremost definitions of functions in variables is that the set of connected statements, values, and codes sorted and keep along to perform a selected task.

On the opposite hand, a collection could be a collective term to visit Python information varieties with changeable free from duplication organized in Associate in Nursing unordered manner. Since you're learning concerning Python variables, understanding functions and sets contributes to familiarising with these terms; therefore, you'll be

able to simply move with Python programming languages. Bear in mind that sets for had provided a selected index order of knowledge whereas functions might not demand any input however entirely dead once exactly required.

Some of the quality options of sets embody lack of a particular order of every worth, number, code, and also the item is exclusive, collections are immutable , and that they provide modifications like addition and deletion. the most advantage of utilizing sets in Python variables is that it optimizes the strategy of checking part is enclosed within the right set or not. Also, it allows you to with we have a tendency to ass or take away some aspects however ought to solely be distinctive and immutable .

Functions build programs a lot of manageable and arranged largely for programmers engaged on difficult tasks. Such tasks are with smaller and standard lumps of codes, therefore increasing readability and reusability. As such, Python offers 3 varieties of functions; Python intrinsic , user-defined and anonymous functions. The functions will solely be at first named, written, and dead to perform a selected reasonably input and build a desired program essential for each beginners and programmers.

Chapter 5

Data Structures

In this chapter, we have a tendency to area unit planning to explore the various knowledge structures in Python.

Sequence

A sequence could be a terribly basic term in Python that's accustomed denote the ordered set of values. There are a unit several sequence knowledge sorts in Python: str, unicode, list, tuple, buffer and xrange.

Tuples

A tuple consists of some values separated by commas. Tuples are a sequence knowledge kind in Python, like strings and lists. we want to stay in mind that tuples area unit changeless. It implies that they can't be modified.

The tuples accommodates the amount of values separated by a comma. The tuples area unit b in parentheses, whereas the lists area unit b in brackets.

Now let's see AN example:

```
>>> m = (14, 34, 56)

>>> m

(14, 34, 56)

>>> m[0]

14

>>> m[ 0:2 ]

(14, 34)
```

Tuples even have the properties like assortment and slicing. Tuples is nested. parts in a very tuple is sorted with ()

Now let's see AN example:

```
i = 1

j = 2

t1 = i, j # could be a tuple consists to parts i and j

t2 = (3, 4, 5) # could be a tuple consists to parts three,4 and 5

t3 = 0, t1, t2 # could be a tuple consists to parts zero, t1 and t2

print t3 # result's (0, (1, 2), (3, 4, 5))
```

Lists

A list consists of some heterogeneous values separated by commas b by [and] and began from index zero. Lists is accustomed cluster alternative values. not like Tuples, Lists area unit mutable . In

alternative words, they will be modified by removing or reassigning existing values. Also, new parts is inserted into the present ones.

Now let's see AN example:

```
>>> a = [1, 2, 3, 4, 5]
>>> a
[1, 2, 3, 4, 5]
```

As strings, lists can even be indexed and sliced.

```
>>> a = [1, 2, 3, 4, 5]
>>> a
[1, 2, 3, 4, 5]
>>> a[0]
1
>>> a[4]
5
>>> a[ 0:2 ]
[1, 2]
>>> a[ 3:5 ]
[4, 5]
```

Unlike strings, lists area unit mutable (i.e. the values is changed)

```
>>> b = [1, 2, 4, 7, 9]
>>> b
[1, 2, 4, 7, 9]
>>> b[2] = six
>>> b
[1, 2, 6, 7, 9] # Here the index [2] is modified to six (the initial worth is 4)
```

>>> b[0] = nine

>>> b

[9, 2, 6, 7, 9]

Here the index [0] is modified to nine (the initial worth is 1)

The values within the list is separated by mistreatment comma (,) between the punctuation. Lists is nested. List is used as a Stack or a Queue.

For example:

list1 = [one, 2, 3, 4]

print len (list1) # returns four - that is that the length of the list

list1[2] # returns three - that is third component within the list Starts

list1[-1] # returns four - that is extreme last component within the list

list1[-2] # returns three - that is extreme last however one component

list1[0:2] = [eleven, twenty two] # commutation 1st 2 parts one and a pair of with eleven and 22

stackList = [one, 2, 3, 4]

stackList.append(5) # inserting five from the last within the stack

print stackList result : [1, 2, 3, 4, 5]

stackList.pop() # removing five from the stack Last In 1st Out

print stackList result : [1, 2, 3, 4]

queueList = [one, 2, 3, 4]

queueList.append(5) # inserting five from the last within the queue

print queueList result : [1, 2, 3, 4, 5]

del(queueList[0]) # removing one from the queue 1st In 1st Out

print queueList result : [2, 3, 4, 5]

Sets

A set doesn’t have ANy duplicate parts gift in it and it's an unordered assortment kind. It means that it'll have all distinct parts in it with no repetition.

Now let’s seen AN example:

fruits = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']

basket = set (fruits) # removed the duplicate component apple

print 'orange' in basket # checking orange in basket, result's True

print 'pineapple' in basket # checking pine apple in basket, result's False

a = set('aioeueoiaeaeiou’) # produce a collection while not duplicates

b = set('bcokcbzo') # produce a collection while not duplicates

print a # a = ['a', 'i', 'e', 'u', 'o']

print b # b = ['z', 'c', 'b', 'k', 'o']

print a & b # letters in each a and b (A ∩ B)

print a | b # letters in either a or b (A ☐ B)

print a - b # letters in a very however not in b (A – B)

Dictionaries

Dictionaries area unit the info structures in Python that area unit indexed by keys.

Key and values separated by “:” and pairs of keys separated by a comma and b by

Lists can't be used as keys.

Now let’s see AN example:

capitals =

capitals['TN'] = 'Chennai'

print capitals['AP']# returns worth of AP within the wordbook

del capitals['TN'] # deletes American state from the wordbook

```
capitals[ 'UP' ] = 'Luck now' # adding UP to the wordbook
print 'AP' in capitals # checks wherever AP key exist in wordbook
print 'TN' in capitals
Numbers =
for key, worth in Numbers.iteritems() :
print key, value
```

Strings

In Python, a string is known by the characters in quotes, like single (‘’) and double (””). they will solely store character values and area unit primitive datatype. Please note that strings area unit altogether completely different from integers or numbers. Therefore, if you declare a string “111”, then it's no relation with the amount 111.

```
>>> print "hello"
hello
>>> print 'good'
good
```

The string index starts from zero in Python.

```
>>> word = 'hello'
>>> word[0]
'h'
>>> word[2]
'l'
```

Indices may additionally be negative numbers, to start out investigation from the correct. Please note that negative indices begin from -1 whereas positive indices begin from zero (since -0 is same as 0).

```
>>> word = 'good'
>>> word[-1]
```

'd'

>>> word[-2]

'o'

The slicing in Python is employed to get substrings, whereas index permits U.S.A. to get one character.

>>> word = 'develop'

>>> word[0:2]

'de'

>>> word[2:4]

've'

Please note that the beginning position is often enclosed and therefore the ending position is often excluded.

Develop

0 one two three four five six ---- Index worth

In the higher than example, the word is assigned a price develop. Considering the primary statement word [0:2], the output is ‘de’. Here the beginning position ‘d’ (0th index) is enclosed and therefore the ending position ‘v’ (2nd index) is excluded. Similarly, within the second statement word [2:4], the beginning position ‘v’ (2nd index) is enclosed and therefore the ending position ‘l’ (4th index) is excluded.

The vital purpose to be noted is that Python strings are immutable (i.e. Strings can not be changed).

There are several in-built functions on the market with a String. they're used for numerous functions. Let’s see a number of the essential ones that are most ordinarily used.

• Len: it's the length operate that's accustomed calculate the amount of characters gift within the string.

• Lower: it'll convert all the great characters gift within the string to minuscule letters. Therefore, once victimisation this operate, all characters within the string are tiny case solely.

• Upper: it'll convert all the minuscule characters gift within the string to great letters. Therefore, once victimisation this operate, all characters within the string are majuscule solely.

• Split: It helps to separate the string into components by employing a delimiter. It is separated victimization areas, new lines, commas, or tabs.

Control Flow Statements

If–else statement

The if-else statement is employed to form the alternatives from two or additional statements. It becomes useful after you wish to execute a selected statement supported a real or False condition.

The syntax of if statement is:

If condition:

action-1 # Indentation

Else:

action-2 # Indentation

Here the indentation is needed. The actions action-1 and action-2 might incorporates several statements however they need to be all indented.

if :

else :

The example is shown below.

```
>>> e = 6

>>> f = 7

>>> if(e < f):

... print( 'f is larger than e' )

... else:

... print(' e is larger than f')
```

...

Output: f is larger than e

```
def numberProperty1 ( input ) :

if computer file two == zero :

print input , ' is an excellent range '

else :

print input , ' is associate Odd range '

numberProperty1( ten ) # result's ten is an excellent range

numberProperty1( eleven ) # result's eleven is associate Odd range
```

Nested If

It consists of over two statements to settle on from.

```
def numberProperty2 ( input ) :

if input < 0:

print input , ' could be a Negative range '

elif input == 0:

print input , ' is Zero '

else:

print input , ' could be a Positive range '

numberProperty2( -100 ) # -100 could be a Negative range

numberProperty2( zero ) # zero is Zero

numberProperty2( one hundred ) # one hundred could be a Positive range
```

While Loop

The whereas loop can run till the expression is true and it stops once it's false.

The syntax of whereas loop is:

```
While expression:
```

statement

For example:

>>> a = 1

>>> while(a < ten):

... print "The range is:" , a

... a = a + 1

The number is: one

The number is: two

The number is: three

The number is: four

The number is: five

The number is: half dozen

The number is: seven

The number is: eight

The number is: nine

The number is: ten

In the on top of example, the block consists of print and increment statements, it's dead repeatedly till the count isn't any longer but five.

def printSeries(begin, end, interval) :

print " \n "

temp = begin

while (worker < finish) :

print temp,

temp += interval

printSeries(one, 11, one) # result's one two three four five half dozen seven eight nine ten

printSeries(one, 11, three) # result's one four seven ten

For Statement

Any object with associate iteration methodology is employed in a for loop in Python. The iteration methodology implies that the information is conferred list type wherever there are multiple values in an ordered manner. The syntax of for loop is:

for item in list:

action # Indentation

The action consists of 1 or additional statements and it should be indented. The examples are shown below.

For example:

for i is {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

... print i

1

2

3

4

5

6

7

8

9

10

>>> list = ['a', 'bb', 'ccc', 'dddd']

>>> for l in list:

... print l,len(l)

...

a 1

bb 2

ccc 3

dddd 4

The on top of program shows that the values of a listing and its length are written victimisation the for loop.

Functions

A operate could be a block of organized and reusable code that's accustomed perform connected tasks. we are able to break our immense lines of programming code into smaller modules with the assistance of functions. It additionally helps in avoiding repetitions of code, as we tend to don't ought to write an equivalent lines of code once more and once more. Instead, we are able to write it once within a operate so use the operate anyplace within the program.

You need to form certain that the operate name is exclusive.

Rules to outline a operate in Python

In Python, a operate is outlined victimisation the keyword def. The arguments are placed among the parenthesis ().

Now let's see associate example:

>>> def printdetails(name, age):

... print "Name:", name;

... print "Age:", age;

... return;

...

>>> print details(name = "Mary", age = 30);

Name: Mary

Age: 30

In the on top of example 'print details 'is the operating name and name and age are the parameters.

Syntax of user-outlined methodology

def < operate name> :

[< declaration of native variables >]

[< statements >]

Now let's see associate example:

```
Language = "Python"
def printString( input ) :
print input
def multiply ( x, y ) :
return x * y
def power( x, y):
return x ** y
printString( Language ) # returns Python
z = multiply( ten, 20 )
print z# returns two hundred - that is up to ten * twenty
print power( ten, two ) # returns one hundred - that is up to ten ** two
```

Accepting inputs throughout the runtime

raw_input() could be a intrinsic Python operate provides the power to simply accept input throughout the execution of the script

Now let's see an associate example:

This statement provides a message to the user to produce input for a reputation.

Control Statements

Break

The break statement breaks out of the littlest inclosure for or whereas loop.

Now let's see associate example:

```
def primeNumberValidation ( input ) :
for x in range( two, input ) :
if computer file x == 0:
print input, 'is not a chief range and equals', x, '*', input/x
break
else:
print input, 'is a chief number'
primeNumberValidation( three )
primeNumberValidation( fourteen )
```

Continue

The continue statement continues with following iteration of the loop.

Now let’s see associate example:

```
def evenNumbers( begin, end ) :
print "\n\nEven characters in-between ',start, 'and', end.
for n in range( begin + one, end ) :
if n they two != 0:
continue
print n
evenNumbers( one, eleven ) # result's fourteen is two four half dozen eight ten
evenNumbers( ten, thirty ) # result's twelve fourteen sixteen eighteen twenty twenty two twenty four twenty six twenty eight
```

Pass

The pass could be a valid statement and may be used once there's an announcement needed syntactically, however the program needs no action.

Now let’s see associate example:

while True :

pass # In condition loop press (Ctrl + c) for the keyboard interrupt

In this example, whereas followed by “pass” it doesn't execute any statement.

There is a necessity to incorporate a minimum of one statement in an exceedingly block (e.g. function, while, for loop, etc.) in these cases, use pass joined statement, that will nothing however includes one statement beneath ‘:’

Now let’s see associate example:

def x() :

pass # one valid statement that doesn't do any action

Here pass is taken into account an announcement for the declaration of operate x.

String Manipulation

We can use intrinsic functions to govern strings in Python. The package “string” provides additional functions on strings.

For example:

print name = "ABCD XYZ xyz"

print len(name) # it'll come the length of the string name

print list(name) # it'll come the list of characters in name only print

name.startswith('A') # it'll come True if name starts with A else returns False

print name.endswith('Z') # it'll come True if name ends with Z else returns False

print name.index('CD') # it'll come the index of CD in name only

print 'C'.isalpha( ) # it'll come True if C is alpha or returns False

print '1'.isdigit( ) # it'll come True if one is digit or returns False

print name.lower( ) # it'll come a string with minuscule characters in name only

print name.upper( ) # it'll come a string with great characters in name only

Exception Handling

Exceptions square measure the errors detected throughout execution and these aren't categorically fatal.

Exception blocks are going to be fenced in with attempt to except statements.

try :

except sort > :

Let’s see AN example:

shaping AN exception block

try:

print (one / zero)

except Exception as excep:

print "exception : ", excep

shaping a user-defined exceptio n

class UserDefinedException(Exception) :

def __init__(self, value):

self.value = price

def __str__(self):

come back repr(self.value)

Raising a user-defined exception expressly

try:

raise UserDefinedException(" input is null ")

except UserDefinedException as userdefinedexception:

print 'userdefinedexception : ', userdefinedexception.value

In the above-named program, first (try, except, block) handles the Zero division exception.

Second (try, except) block raises a user-defined exception.

Chapter 6

Learning concerning Functions

So far, we've learned quite an heap of things. If you've got already began to lose track of all the information, you shouldn't be afraid. it's solely natural for everybody to seek out themselves in such a scenario after they square measure within the learning method. nobody is ideal which is what makes North American nation all people in general, right?

We have seen dictionaries and learned they're nothing just like the ones we tend to use to be told new words and meanings. we've learned a couple of rather funny factor referred to as tuples and understood that they're primarily an inventory with parentheses and don't enable anyone to feature, remove, or modify values. we've gone at first through some functions too, however currently it's time for North American nation to begin trying into functions a bit additional closely.

Understanding the conception of operate

Take an instant or 2 here and interact your mind a bit. accept it and check out to return up with some imprecise plan of what functions really square measure.

Functions square measure either user-defined or pre-defined. In either case, their job is to arrange codes at intervals a recallable operate name. There square measure loads of pre-defined functions out there at intervals Python. we've already been victimization a number of these once more and once more.

We have already got an honest plan concerning functions that square measure integral and pre-defined. These embody and aren't restricted to input(), print(), and then more. However, let's currently consider the way to produce our operate.

Let's begin by a conventional approach and write a block of code that welcomes the user with a friendly acknowledgement. we are

going to store this as a operate named “welcome_message” in order that we will invoke this operate afterward.

```
def welcome_message():

print("Hello and welcome")

print("Hope you've got an excellent time")

print("Begin")

welcome_message()

print("End")
```

Let’s begin learning and see what's happening within the block of code higher than. Firstly, for North American nation to make our operate, we'd like to outline it initial. The ‘def’ could be a keyword that Python can consider and in real time perceive that you just square measure getting ready to ‘define’ a brand new operate. Next, we are going to got to name the operate. whereas you'll invariably name the operate as you please, it's extremely suggested and inspired that you just use names that square measure straightforward to grasp and have a descriptive name. If we tend to were to call this operate something aside from welcome_message, we tend to might recognize what it's as we tend to wrote it, except for the other applied scientist out there, they'll not perceive.

Whenever you produce a operate, you wish to use parentheses. you are doing not need to pass any data through it thus leave them as they're. Now, we'd like to feature the colon mark.

What happens after you use a colon at the tip of a statement? Your indicator gets indented within the following line. meaning your indicator are going to be slightly aloof from the particular place to begin. this can be to denote to the applied scientist that he/she is getting ready to sort one thing that might hold price for a command or an announcement higher than it. during this case, we tend to try to outline the operate.

Let’s then use 2 print commands and place our acknowledgement messages. that's it! you've got currently created your terribly initial operate. you'll currently recollect it as persistently as you wish.

However, must you try and invoke this operate a line or 2 before the ‘def’ command, Python can don't have any plan what you’re talking concerning. Why? That has everything to try and do with the very fact that Python reads a command by line. By the time it arrives on the road wherever you referred to as a operate, it'd visit the previous lines and not realize something relatable because the actual ‘def’ step was meted out in a very step following this.

After this, let’s currently use our operate and see however it works. Remember, the operate holds 2 printable messages for our users. For our reference, we are going to currently produce a ‘begin’ ANd an ‘end’ message. this may enable North American nation and also the applied scientist to grasp wherever the regular messages square measure and wherever the operate lies. Use your operate with empty parentheses between the 2 print commands as shown higher than. If you wish, you'll take away these print commands and simply sort in your operate range to ascertain the results.

A quick tip for all! If you come upon the annoying wiggly lines, merely hover your mouse over it and you'll ascertain what the expected or urged resolution is. during this case, if you take away the two-line areas, you ought to see a suggestion voice communication this:

Whenever you outline a operate, you'll invariably be needed to depart a minimum of 2 blank lines before continuing on with the codes.

Now, let’s run the program and you ought to see all the messages and our operate in action. Python initiated the sequence and initial scan the definition. this can be wherever Python solely understood for itself what the operate was. the particular program was dead once Python reached line six, wherever our print(“Begin”) message started. within the next line, we tend to placed our operate and this can be wherever Python recalled what it had simply learned. It quickly meted out the set of codes we tend to outlined at intervals the operate and dead an equivalent. Lastly, it dead the last line before finishing the program.

This is however functions square measure created and used. Now, we will use this operate as persistently as we tend to like at intervals

an equivalent file. Note that you just cannot use this new created operate if you were to open a brand new file or work on AN older file wherever you probably did not outline this operate.

When things begin to induce more durable for you in your programming future, bear in mind to make your functions and use them wherever applicable. they're going to prevent quite an heap of your time and assist you in places in addition. These square measure used once sure actions or operations got to be meted out each currently so.

Using numerous Functions

Python was created with simplicity in mind. it absolutely was additionally created to attenuate the work and maximize the output. If you utilize the codes and also the functions sagely, you'll sure as shooting be creating the foremost out of this artificial language. it's additionally noticeable that the majority of the items you study Python and its functions, parameters, methods, and such can assist you learn different languages faster, thus do concentrate.

Parameters

Our farsighted readers might have noticed one thing concerning the operate we tend to simply created a couple of minutes a gone. not like most of the functions, we tend to didn't pass any data through the parentheses in the slightest degree. Why that happens are a few things come back concerning once we tend to understand precisely what parameters are in Python.

Parameters square measure used as place-holders for receiving data. These square measure what we tend to, in addition as users, offer to the program so as for it to figure additional accurately. There square measure some cases and functions wherever arguments aren't needed for them to try and do their basic operation. However, if you offer AN argument to those functions, they're going to offer you with a additional specific output. Of course, it will rely on the provision of the aforesaid parameter. you can't force a operate to try and do one thing it's not designed to try and do.

Now, let’s consider our operate. it's actually missing one thing. If you presently print the welcome_user operate, it'd say everything however won't contain the name of the user in the slightest degree. Surely, it'd look heaps nicer for North American nation if we tend to might somehow use this operate to use the name of the user and add it to the acknowledgement.

Luckily, we will just do that! For that, we tend to initial got to add the ‘name’ parameter within the initial line, wherever we tend to begin shaping our operate. straightforward sort name between the parentheses and you'll see the text flip gray. This confirms that the word has been another as a parameter. Now, we tend to would like to print the name of this user together with the greetings we've outlined at intervals the operate. For this instance, let’s assume that the user is known as Fred.

def welcome_message(name):

print("Begin")

print("Hello and welcome !")

print("Hope you've got an excellent time")

print("End")

welcome_message('Fred')

Begin

Hello and welcome Fred!

Hope you've got an excellent time

End

Finally! we've a reputation to feature to our greetings. you'll add another line of code by victimization our operate and spending a unique name currently. See what happens then.

When we set a parameter for a operate so decision upon the operate while not providing it with AN argument or the little bit of data that goes between the parentheses, it'll offer North American nation with a slip-up, apart from a couple of.

Now, let’s build our operate a bit additional dynamic and add another parameter. Let’s add a parameter that permits the program to print out the cognomen of the user. Now, our code ought to look one thing like this:

```
def welcome_message(name, last_name):

print("Hello and welcome !")

print("Hope you've got an excellent time")

print("Begin")

welcome_message('Fred', 'William')

print("End")
```

The point to be told here, excluding having the ability to feature parameters, is that the indisputable fact that ‘Fred’ and ‘William’ square measure getting used in a very specific order. must you sort it the opposite method around, Python can print these as they're. this can be as a result of their position regarding the outlined parameters. the primary price Python reads here, it'll mechanically link it with the primary parameter. this will cause a bit confusion, particularly if the cognomen becomes the primary name.

These arguments are known as point arguments. To additional show their importance, let’s take away the cognomen from the argument on top of.

```
print("Begin")

welcome_message('Fred')

print("End")
```

```
Traceback (most recent decision last):

Begin

File "C:/Users/Smith/PycharmProjects/MyFirstGo/PosArg.py", line 7, in

welcome_message('Fred')
```

TypeError: welcome_message() missing one needed point argument: 'last_name'

So, the system doesn't enable North American country to continue as we've got removed a component. This time, sort within the cognomen 1st followed by the primary name and see if it makes any distinction. after you run the program currently, you must be ready to see this:

print("Begin")

welcome_message('Fred', 'William')

print("End")

Begin

Hello and welcome William Fred!

Hope you have got an excellent time

End

Now, the sequence quite worked. the sole issue is that it's gotten the name wrong. Now, the cognomen is being represented and written because the forename. that's rather embarrassing, isn’t it?

The on top of errors either state that we have a tendency to are missing one needed point argument or show that we have a tendency to placed the incorrect name within the wrong place. point arguments are such arguments whose position matters tons. If you miss out on the position altogether, you may find yourself with a slip. If you sort in one thing else, as we have a tendency to did in our last example, you may manufacture incorrect results. To correct it, merely give the cognomen once the primary name.

There is an added means you'll have these restricted by mistreatment what's termed as ‘keyword arguments’. These are the sort of arguments whose position doesn't matter in any respect and Python can still still operate properly notwithstanding their position within the parentheses. To pass a keyword argument, you may have to be compelled to do the following:

print("Begin")

```
welcome_message(last_name='William', name='Fred')
print("End")
```

```
Begin
Hello and welcome Fred William!
Hope you have got an excellent time
End
```

Now that’s additional am passionate about it. Things are trying right at however we wish them. Notice how, albeit we have a tendency to wrote within the wrong order, Python picked up and sorted the order for North American country. that's as a result of we have a tendency to created our entries or arguments into keyword arguments mistreatment the name= or last name= parameter and mixing it with arguments. this permits Python to draw info and perceive that of those 2 comes 1st so as outlined originally in our operate.

Factually speaking, you may not be mistreatment these quite ton, however it's continually a plus to grasp the ways in which to beat sure problems. Normally, you or the other coder would be ready to see the information and browse simply if you just follow the principles and sort forename followed by cognomen. build the code as straightforward as you'll for everybody to browse and perceive.

That is one fine question. this is often wherever you may have to be compelled to use keyword arguments to represent what those values are for. you would possibly be running a operate that involves multiple values that solely you would possibly be ready to perceive, however others can don't have any plan wherever they came from. you need to label each of them with the relevant keyword argument so the readability will increase.

We are presently simply starting and for the sake of demonstration, we have a tendency to use a straightforward example to showcase the way to produce functions and use them. Your functions, once the time comes, may be quite huge or equally short, looking on the sort of operate you produce of any specific scenario.

Creating functions actually helps North American country organize our codes and be additional economical and effective. If we have a tendency to were unable to try to this, we'd have had to resort to writing constant bunch of lines each currently then.

Return Statement

We might have lined what a comeback statement is after we were discussing ‘if’ statements et al.. However, it makes additional sense to be told this once you have got understood the idea of parameters and functions.

So far, we've got created a operate that has allowed North American country to send info via the utilization of parameters. However, after we remark come back statements, these are designed to try to sure calculations and supply North American country with the results rather than North American country feeding it with values.

Let’s look a bit deeper into however this works by making our second operate. the aim of this operate relies on a straightforward maths trick that most folks may need detected of or vie with after we were young. raise the user to consider any variety and you'd raise them to feature and work out some straightforward numbers. Eventually, you'd give them with associate degree correct result and everybody would be dismayed. Now, let’s reveal what happens with the utilization of this operate we have a tendency to are near to produce.

def magic_number(number):

return variety + half dozen - four + five - variety

This simple calculation would continually come back you the worth of seven. Go ahead, attempt it out yourself by doing this. However, for North American country to be ready to get these values back, we've got used the comeback statement here. This tells Python that it's presupposed to do the calculation for North American country then solely come back the ensuing price rather than printing every of those singly.

Let’s provide our second operate a check run:

result = magic_number(8329)

print(result)

See however the result currently shows as seven? you'll try to amendment the values to no matter you please, the result can still stay as seven. that's solely created doable due to the come back statement we've got provided here. If you're taking away the keyword come back, you finish up with a worth that claims ‘None’. The program can still operate, however the result would not be calculated or of any North American country to us. this is often as a result of Python wouldn't execute a comeback part and therefore won't do the calculations as we'd am passionate about it to.

Using these will greatly enhance your expertise as a coder or a user. However, before you dive in and begin making your functions, here are some that are pre-defined and should are available in handy. there's no purpose in making a operate and looking for Python already had one for you.

1. min() and max() - just in case you run into numerous prices and you quickly would like to search out the minimum value existing among a listing or assortment of information, use the min() command and run the numbers through. The minimum variety are written for you. The max() operate is that the opposite, of course!

2. sum() - this is often quite keen operate and permits you to quickly add up all the numbers within the list and manufacture the result for you directly. The accuracy with floats won't be what we have a tendency to like, but hey, it gets you going.

3. type() - There might return lines and features of codes with variables that are scattered everywhere the place. You currently would like to search out wherever the variable started from and what quite a variable it's. mistreatment the type() operate, you'll quickly conclude what quite variable you're addressing. it'll come back values like ‘bool’ to point that the variable in question may be a bool in sort.

There are many functions that you simply can begin to be told as you proceed into advanced Python learning and machine learning. However, to grasp most of them, you may have to be compelled to

apply these and develop an intensive understanding of however this works. you must then don't have any issues venturing into a additional advanced version of Python learning and developing complicated programs.

Chapter 7

Conditional and Loops in Python

This chapter describes moderate level topics like conditionals and loops well. we'll use completely different examples to clarify these topics well. Let’s dive into knowing a lot of regarding these ideas.

What is a sequence in Python?

The sequence of program execution isn't a road linking the north and also the south. It will run from the north to the south to the top. The sequence of program execution is also as difficult as a road within the busy space, with 9 turns and eighteen turns, that is straightforward to create folks dizzy.

To write an honest program, it's vital to regulate the method of program execution. Therefore, it's necessary to use the method management structure of the program. while not them, it's not possible to use the program to finish any difficult work.

The programing language has been unendingly developed for many years. Structured Programming has bit by bit become the thought of program development. Its main plan is to execute the whole program in sequence from high to bottom. Python language is principally dead from high to bottom consistent with the sequence of program ASCII text file, however typically the execution sequence are modified consistent with wants.

At now, the pc are often told that sequence to execute the program preferentially through flow management directions. the method management of the program is like coming up with a traffic direction extending altogether directions for the transportation.

It is recognized that almost all program codes for method management are dead in sequence from high to bottom line when line, except for operations with high repeatability, it's not appropriate to execute in sequence. Any Python program, regardless of however complicated its structure is, are often expressed or delineated

mistreatment 3 basic management processes: sequence structure, choice structure, and loop structure.

The first line statement of the sequence structure program is that the entry purpose and is dead from high to bottom to the last line statement of the program. the choice structure permits the program to pick the program block to be dead consistent with whether or not the check condition is established or not. If the condition is True, some program statements are dead. If the condition is fake, alternative program statements are dead.

Colloquially, if you encounter a scenario A, perform operation A; if this can be case b, operation b is dead. rather like after we drive to the intersection and see the signal lamp, the red light-weight can stop, and also the inexperienced light-weight can pass. Also, destinations even have different directions, and you'll be able to opt for the route consistent with completely different things. In alternative words, the choice structure represents that the program can confirm the "direction" of the program consistent with the required conditions.

The perform of loop flow management with loop structure is to repeatedly execute the program statements during a program block till the particular ending conditions are met. Python includes a for loop and a jiffy loop.

Selection method management

Selection method management may be a conditional management statement that contains a conditional judgment also brought up as conditional expression or conditional judgment expression for short). If the results of the conditional judgment expression is True (true), a program block is dead. If the results of the conditional judgment expression is fake (True), another program block is dead.

The following describes the statements and their functions associated with the choice method management in Python language.

If...Else Conditional Statement

If...else conditional statement may be a fairly common and sensible statement. If the conditional judgment expression is True (true, or described by 1), the program statement within the if program block is

dead. If the conditional judgment expression isn't true (False, or described by 0), the program statement within the else program block is dead. If there are multiple judgments, elif instruction are often another.

The syntax of the if the conditional statement is:

If the conditional judgment expression holds, execute the program statement during this program block

Else ：

If the condition doesn't hold, execute the program statement during this program block. If we wish to guage whether or not the worth of variable a is bigger than or adequate to the worth of variable b, the condition judgment expression are often written as follows:

If a >= b:

If A is bigger than or adequate to B, execute the program statement during this program block

Else ：

If a isn't larger than or adequate to b, the program statement if ... if…else conditional statement during this program block is dead.

In the use of the if...else conditional statement, if the condition isn't happy, there's no ought to execute any program statement, and also the else half are often omitted

If conditional judgment expression

If the condition is happy, execute the program statements during this program block. Besides, if the if…else conditional statement uses logical operators like “and”, it's instructed to feature parentheses to differentiate the execution order to enhance the readability of the program,

For example: if (a==c) and (a>b):

If A equals C and A is bigger than B, execute the program statement during this program block

Else ：

If the on top of condition doesn't hold, the program statement during this program block is dead.

Also, Python language provides a lot of elliptic conditional expression of if...else within the following format: X if C else Y returns one in every of the 2 expressions consistent with the conditional judgment expression. within the on top of expression, X is came back once C is true; otherwise, Y is came back.

For example, to see whether or not the number x is odd or perhaps, the initial program would be written as follows:

If (first nada 2)==0:

second= "even number"

Else:

second= "odd number"

If print(".format(second)) is modified to a elliptic type, solely one line of program statements is needed to realize an equivalent purpose.

The statements are as follows:

print(".format ("even" if (first% 2)==0 else "odd"))

If the if condition determines that the expression is true, it returns "even"; otherwise, it returns "odd." within the following sample program, we'll follow the employment of the if…else statement. the aim of the sample program is to create a straightforward intercalary year judgment program.

Let the user enter the year (4-digit number year), and also the program can confirm whether or not it's a intercalary year. one in every of the subsequent 2 conditions may be a leap year:

(1) leap each four years (divisible by 4) however not each one hundred years (divisible by 100).

(2) leap each four hundred years (divisible by 400).

[example procedure: leapYear.py]

Determine whether or not Associate in Nursing input year may be a intercalary year or not

01 # -*- coding: utf-8 -*-

02 """

03 program name: intercalary year decision making program

04 Topic Requirements:

05 Enter the year (4-digit number year) to see whether or not it's a intercalary year

06 condition one. each four leap (divisible by 4) and each one hundred leap (divisible by 100)

07 condition a pair of. each four hundred leap (divisible by 400)

08 one in each of the 2 conditions met maybe an intercalary year.

09 """

10 year = int(input ("Give year:"))

12 if (year to 1 / 4 == zero and years one hundred ! = 0) or (year nada four hundred == 0):

13 print(" may be a intercalary year ."format(year))

14 Else:

Program Code Resolution:

Line 10: Enter a year, however bear in mind to decision the int () operate to convert it to Associate in Nursing whole number sort.

Line 12-15: choose whether or not it's a intercalary year.

Condition 1: each four leaps (divisible by 4) and each one hundred leaps (not dissociative by 100).

Condition 2: each four hundred leaps (divisible by 400). one in all the 2 conditions may be a intercalary year. Readers are asked to inquire whether or not the subsequent years are leap years: 1900 (flat year), 1996 (leap year), 2004 (leap year), 2017 (flat year), 2400 (leap year).

Multiple decisions

If there's over one conditional judgment expression, elif conditional statement may be additional. Elif is just like the abbreviation of "else if." though exploitation multiple if conditional statements will solve the matter of execution completely different program blocks beneath numerous conditions, it's still not straightforward enough. Then, elif conditional statements may be used, and also the readability of the program may be improved.

Note that if the statement may be a logical "necessity" in our program. Elif and else don't essentially follow, therefore there are 3 situations: if, if/else, if/elif/else.

The format is as follows:

If condition judgment

Expression 1:

If the conditional judgment expression one holds, the program statement during this program block is dead

Elif condition judgment

Expression 2:

If the conditional judgment expression two holds, execute the program statement during this program block

Else ：

If none of the on top of conditions hold, execute the program statement during this program block,

For example:

If first==second:

If initial equals second, execute the program statement during this program block

Elif first>second ：

If initial is larger than second, execute the program statement during this program block

Else ：

if initial isn't up to second and initial is a smaller amount than second, execute the program statement during this program block.

The following example program is employed to follow the employment of IF multiple choice. the aim of the sample program is to discover the present time to come to a decision that salutation to use.

[sample procedure: currentTime.py]

Detects the present time to come to a decision that salutation

```
01 # -*- coding: utf-8 -*-
02 """
03 Program Name: discover the present time to come to a decision that salutation to use
04 Topic Requirements:
05 decision making from the present time (24-hour system)
06 5~10:59, output "good morning"
07 11~17:59, output "good afternoon"
08 18~4:59, output "good night"
09 """
11 import time
13 print ("current time: .” format (time.strftime ("%h:% m:% s"))
14 h = int( time.strftime("%H") )
16 if h>5 and h < 11:
17 print ("good morning!" )
18 elif h >= eleven and h<18:
19 print ("good afternoon!" )
20 else:
21 print ("good night!")
```

The execution results of the program are shown on the screen.

The output shows the present time within the sample program to guage whether or not it's morning, afternoon, or evening, so displays the acceptable salutation. Python's time module provides numerous functions associated with time. The Time module may be a module in Python's commonplace module library.

Before exploitation it, you wish to use the import instruction to import so decision the strftime operate to format the time into the format we have a tendency to wish. as an example, the subsequent program statement is employed to get the present time.

Import time

Time.strftime ("%h:% m:% s")

18: 36: sixteen (6:36:16 p.m. 24-hour)

Time. strftime ("%i:% m:% s")

06:36:16 (6: 36: sixteen p.m. 12-hour system) format parameters to be set are capsulate in parentheses.

Pay attention to the case of format symbols. the subsequent program statement is employed to show the week, month, day, hour, minute, and second.

Print (time.strftime ("%a,% b% usual h:% m:% s")) execution results are as follows: Mon, sep17 15: four9: twenty nine 4.2.3 nested if generally there's another layer of if conditional statement within the if conditional statement. This multi-layer choice structure is termed nested if conditional statement.

Usually, once demonstrating the employment of nested if conditional statements, it's additional common to demonstrate multiple decisions with numerical ranges or scores. In alternative words, completely different grades of certificates are issued for various grades of achievements.

If it's over sixty points, the primary certificate of competence are given, if it's over seventy points, the second certificate of competence are given, if it's over eighty points, the third certificate of

competence are given, if it's over ninety points, the fourth certificate of competence are given, if it's over one hundred points, the all-around skilled certificate of competence are given.

Based on nested if statements, we will write the subsequent program:

Available= int(input ("Give a score:")

if on the market >= 60:

print ('First Certificate of Conformity')

if on the market >= 70:

print ('Second Certificate of Conformity')

if on the market >= 80:

print ('Third Certificate of Conformity')

if on the market >= 90:

print ('Fourth Certificate of Conformity')

if getScore == 100:

Print ('All-round skilled Qualification Certificate') is really Associate in Nursing if statement that's explored layer by layer. we will use the if/elif statement to filter the multiple decisions one by one per conditional expression operation and choose the matching condition (True) to execute the program statement in an exceedingly program block.

The syntax is as follows:

If Conditional Expression 1:

The program block to be dead beneath conditional expression one

Elif conditional expression 2:

The program block to be dead beneath conditional expression two

Elif conditional expression n:

The program block to be dead per the conditional expression n

Else:

If all the conditional expressions don't change, this program block is dead. once the conditional expression one doesn't change, the program block searches right down to the finally conformist conditional expression.

The elif instruction is Associate in Nursing abbreviation of else if. Elif statement will generate multiple statements per the operation of a conditional expression, and its conditional expression should be followed by a colon, that indicates that the subsequent program blocks meet this conditional expression and wish to be indented.

The following example program may be a typical example of the combined use of nested if and if/elif statements. This program uses if to see that grade the question results belong to. Also, another judgment has been additional to the sample program. If the score whole number price entered isn't between zero and one hundred, a prompt message of "input error, the amount entered should be between zero and 100" are output.

Comprehensive use of nested if statements example:

```
01 # -*- coding: utf-8 -*-

02 """

03 samples of Comprehensive Use of Nested if Statements

04 """

05 result = int(input (‘Give final grade:')

06

07 # initial Level if/else Statement: choose whether or not the result entered is between zero and one hundred

08 if result >= zero and result <= 100:

09 # second level if/elif/else statement

10 if result <60:
```

11 print(' below cannot acquire certificate of competency'. format(result))

12 elif result >= sixty and result <70:

13 print(' result's d'. format(result))

14 elif result >= seventy and result <80:

15 print(' result's c'. format(result))

16 elif result >= eighty and result <90:

17 print(' result's level b'. format(result))

18 else:

19 print(' result's grade a'. format(result))

20 else:

The execution results of the

21 print ('input error, input variety should be between 0-100')

Program code analysis:

Lines 7-21: first-level if/else statement, wont to choose whether or not the input result's between zero and one hundred.

Lines 10-19: the second-level if/elif/else statement, that is employed to guage that grade the inquired result belongs to.

In the next section, we'll discuss loops one in all the foremost necessary ideas.

The Loop Repeat Structure

This principally refers to the loop management structure. an exact program statement is repeatedly dead per the set conditions, and also the loop won't seem till the condition judgment isn't established. In short, repetitive structures are wont to style program blocks that require to be dead repeatedly, that is, to create program code change to the spirit of structured style.

For example, if you wish the pc to calculate the worth of 1+2+3+4+…+10, you do not want US to accumulate from one to ten within the

program code, that is originally tedious and repetitive, and you'll simply reach the goal by exploitation the loop management structure. Python contains a minute loop and a for loop, and also the connected usage is delineated below.

While loop

If the amount of loops to be dead is set, then exploitation the for loop statement is that the best option. However, the whereas loop is additional appropriate sure as shooting cycles that can't be determined. The whereas loop statement is comparable to the for loop statement and belongs to the pre-test loop. The operating model of the pre-test loop is that the loop condition judgment expression should be checked at the start of the loop program block.

When the judgment expression result's true, the program statements within the loop block are dead. we have a tendency to typically decision the program statements within the loop block the “loop body.” whereas loop conjointly uses a conditional expression to evaluate whether or not it's true or false to manage the loop flow. once the conditional expression is true, the program statement within the loop are dead. once the conditional expression is fake, the program flow can stand out of the loop.

The format of the whereas loop statement is as follows:

While conditional expression:

If the conditional expression holds, the flow chart of corporal punishment the whereas loop statement during this program block.

The whereas loop should embrace the initial price of the management variable and also the expression for increasing or decreasing. once writing the loop program, it should check whether or not the condition for going away the loop exists. If the condition doesn't exist, the loop body are unendingly dead no end, leading to associate degree "infinite loop," conjointly referred to as "dead loop."

The loop structure typically needs 3 conditions:

(1) The initial price of the loop variable.

(2) Cyclic conditional expression.

(3) alter the rise or decrease the worth of cyclic variables.

For example, the subsequent procedure:

first=1

While 1st < 10: # Loop Condition Expression

print(first)

first += one # adjusts the rise or decrease price of the loop variable.

When 1st is a smaller amount than ten, the program statement within the whereas loop are dead, then 1st are supplemental with one till 1st is capable ten. If the results of the conditional expression is fake, it'll stand out of the loop.

For loop

For loop, conjointly called count loop, may be a loop type usually utilized in programming. It will repeatedly execute a hard and fast variety of loops. If the amount of loop executions needed is thought to be mounted once coming up with the program, then the for-loop statement is that the best option. The for loop in Python language are often accustomed traverse parts or table things of any sequence. The sequence are often tuples, lists or strings, that area unit dead in sequence.

The syntax is as follows:

For component variable in sequence:

dead directions

Else:

The program block of #else are often supplemental or not supplemental, that is, once mistreatment the for loop, the else statement are often supplemental or not supplemental. The which means pictured by the on top of Python syntax is that the for loop traverses all parts in an exceedingly sequence, like a string or a listing, within the order of the weather within the current sequence (item, or table item).

For example, the subsequent variable values will all be used as traversal sequence parts of a

first= "abcdefghijklmnopqrstuvwxyz "

second= ['january', 'march', 'may', 'july', 'august', 'october', 'december']

result= [a, e, 3, 4, 5, j, 7, 8, 9, 10]

Besides, if you wish to calculate the amount of times a loop is dead, you want to set the initial price of the loop, the ending condition, and also the increase or decrease price of the loop variable for every loop dead within the for-loop management statement. For loop each spherical, if the rise or decrease price isn't specifically nominative, it'll mechanically accumulate one till the condition is met.

For example, the subsequent statement may be a tuple (11 ~ 15) and uses the for loop to print out the numeric parts within the tuple: x = [11, 12, 13, 14, 15]

for 1st in x:

print(first)

A lot of economical thanks to write tuples is to decision the vary () operate directly. The format of the vary () operate is as follows:

range ([initial value], final price [,increase or decrease value])

Tuples begin from "initial price" to the previous variety of "final value.” If no initial price is nominative, the default price is 0; if no increase or decrease price is nominative, the default increment is one.

An example of occupation the vary () operate is as follows: vary (3) implies that ranging from the subscript price of zero, three parts area unit output, i.e., 0, one and a pair of area unit 3 parts in total.

Range(1,6) suggests that ranging from subscript price one and ending before subscript price 6-1, that is, subscript variety vi isn't enclosed, i.e., 1, 2, 3, four and five area unit 5 parts. •range (4,10,2) suggests that ranging from subscript price four and ending before subscript variety ten, that is, subscript variety ten is excluded, and

also the increment price is a pair of, i.e., 4, vi and eight area unit 3 parts. the subsequent program code demonstrates the employment of the vary () operate in an exceedingly for loop to output even numbers between a pair of and eleven for i in vary (2, 11, 2).

One more issue to pay special attention to once mistreatment the for loop is that the print () operate. If the print () is indented, it implies that the operation to be dead within the for loop are output in step with the amount of times the loop is dead. If there's no indentation, it suggests that it's not within the for loop, and solely the ultimate result are output.

We know that occupation the vary () operate with the for loop cannot solely do accumulation operations however conjointly do a lot of varied accumulation operations with the parameters of the vary () operate. as an example, add up all multiples of five inside a definite vary. the subsequent sample program can demonstrate a way to use the for loop to accumulate multiples of five inside a variety of numbers.

[Example Procedure: addition.py]

Accumulate multiples of five in an exceedingly bound numerical vary

```
01 # -*- coding: utf-8 -*-
02 """
03 Accumulate multiples of five inside a definite numerical vary
04 """
05 addition = zero # stores the accumulated result
06
07 # enters for/in loop
08 for count in range(0, 21, 5):
09 addition += count # adds up the values
11 print('5 times additive result =',addition)
# Output additive result
```

Program code analysis:

Lines 08 and 09: Add up the numbers five, 10, 15 and 20. Also, once corporal punishment a for loop, if you wish to grasp the subscript price of a component, you'll decision Python's inherent enumerate operate. The syntax format of the decision is as follows: for subscript price, component variable in enumerate (sequence element).

For example (refer to sample program enumerate. py):

names = ["ram," "raju," "ravi"]

for index, x in enumerate(names):

The execution results of the on top of statement in print ("-." format (index, x)) is displayed.

Nested loop

Next, we'll introduce a for nested loop, that is, multiple for loop structures. within the nested for loop structure, the execution method should expect the inner loop to finish before continued to execute the outer loop layer by layer.

The double nested for loop construction format:

For example, a table is often simply completed employing a double nested for loop. Let's take a glance at a way to use the double nested for loop to form the 9 tables through the subsequent sample program.

[Example Procedure: 99Table.py]

99 Table

01 # -*- coding: utf-8 -*-

02 """

03 Program Name: Table

04 """

05

06 for x in range(6,68):

07 for y in range(1, 9):

08 print("*=.”format(y, x, x * y), end=" ")

99 may be a terribly classic example of nested loops. If readers have learned different programming languages, i think they'll be astonied at the brevity of Python. From this instance program, we are able to clearly perceive however nested loops work. hereunder, the outer layer for the loop is stated because the x loop, and also the inner layer for loop is stated because the y loop.

When coming into the x loop, x=1. once the y loop is dead from one to nine, it'll come to the x loop to continue execution. The print statement within the y loop won't wrap. The print () statement within the outer x loop won't wrap till the y loop is dead and leaves the y loop. once the execution is completed, the primary row of 9 tables are obtained. once all X cycles area unit completed, the table is completed.

Note that the common mistake for beginners is that the sentences of the inner and outer loops area unit staggered. within the structure of multiple nested loops, the inner and outer loops can not be staggered; otherwise, errors are caused.

The continue instruction and break instruction area unit the 2 loop statements we have a tendency to introduced before. below traditional circumstances, the whereas loop is to evaluate the condition of the loop before coming into the loop body. If the condition isn't glad, it'll leave the loop, whereas for loop ends the execution of the loop on balance the desired parts area unit fetched. However, the loop may also be interrupted by continue or break. the most purpose of break instruction is to leap out of the present loop body, rather like its English which means, break suggests that "interrupt."

If you wish to depart the present loop body below the desired conditions within the loop body, you wish to use the break instruction, whose operate is to leap off the present for or whereas loop body and provides the management of program execution to consequent line of program statements outside the loop body. In

different words, the break instruction is employed to interrupt the execution of the present loop and jump directly out of the present loop.

Chapter 8

Inheritance and Polymorphism

In Python, a category is defined using the keyword Class, an equivalent way a function is defined with the keyword def. Therefore, if we were to define a category called ClassName, our syntax would appear as if this:

class ClassName:

“Class Documentation String”

Class_Suite

In this syntax, the keyword class is employed to define a replacement class, followed by the category name ClassName, and a colon. the category documentation string is actually a definition or description of the category . The Class_Suite is representative of everything of the defining class members, component statements, functions, and data attributes of the category .

Since a category is employed to make objects, it's best seen as how to feature consistency to programs created in Python in order that they're cleaner, more efficient, and most significantly , functional. to make a category which will be instantiated anywhere in your code, it must be defined at a module’s top-level.

Creating a category in Python

Now that we are conversant in the syntax for creating a category in Python, we’ll introduce a category example called Dog.

class Dog:

“Dog class”

var1 = “Bark”

var2 = “Jump”

Class Declaration and Definition

In Python 3, there's no difference between class declaration and sophistication definition. this is often because the 2 occur simultaneously. Class definition follows declaration and documentation string as demonstrated in our example.

Class Methods and Attributes

A class, defined and created, isn't complete unless it's some functionality. Functionalities during a class are defined by setting their attributes, which are best seen as containers for functions and data associated with those attributes.

Class attributes include data members like variables and instance variables and methods found using the dot notation. Here are their definitions:

- Class variable: this is often a variable shared by all the category instances and objects.
- Instance variable: this is often a variable unique to an instance of a category . it's typically defined within a way and can only be applicable to the instance of that class.
- Method: Also called a function, a way is defined during a class and defines the behavior of an object.

Class Attributes

A class attribute may be a functional element that belongs to a different object and accessed via dotted-attribute notation. In Python, complex numbers have data attributes while lists and dictionaries have functional attributes. once you access attribute, you'll also access an object which will have attributes of its own.

Class attributes are linked to the classes during which they're defined. the foremost commonly used objects in OOP are instance objects. Instance data attributes are the first data attributes that are used. You'll find most use for sophistication data attributes once you require a knowledge type, which doesn't need any instances.

Class Data Attributes

Data attributes are the category variables which are defined by the programmer. they will be used like all other variable when creating

the category . Methods can update them within the category . Programmers know these sorts of attributes better as static members, class variables, or static data. They represent data tied to the category object they belong to and are independent of any class instances.

Example of using class data attributes (xyz):

```
class ABC:

xyz = 10

print(ABC.xyz)

ABC.xyz = ABC.xyz + 1

print (ABC.xyz)
```

The output of this instance code are going to be 11.

Python Class Inheritance

Inheritance may be a feature in object-oriented programming that permits a category to inherit methods and attributes from a parent class, also mentioned as base class. this is often a really handy feature in programming because it allows the programmer to make a set of functionality for one class then pass them on to sub-classes or child classes. As a result, the program are going to be ready to create new and even overwrite existing functionalities during a child class without affecting the functionality of the parent class.

The sub-classes that inheritance creates will feature the specializations of the parent classes. There are four sorts of inheritances in Python: single, multilevel, hierarchical, and multiple inheritances.

Single Inheritance

Programming classes that haven't any inheritance features are often accurately mentioned as object-based programming. The program, when run, should create new abstract data types, each with its own operations. However, what separates object-oriented programming from object-based programming is inheritance. Single inheritance is

when a category or subclass inherits methods and attributes from one parent class.

Multiple Inheritance

In multiple inheritance, a toddler class or subclass inherits methods and attributes from multiple classes. as an example , a category C can inherit the features of both class A and B, within the same way that a toddler inherits the characteristics of both the mother and father. In some cases, a toddler class can inherit the features and functionalities of quite two parent classes.

There is no limit to the amount of parent classes from which a toddler class can inherit methods and attributes. Note that while multiple inheritance is best known to scale back program redundancy, it's going to also introduce a better level of complexity and ambiguity to the program and must be properly thought-out during program design before implementation.

Multilevel Inheritance

We have already established that it's possible for a category in Python to inherit features of multiple parent classes. When a category inherits the methods and functions of other classes that also inherit them from other classes, the method is understood as multilevel inheritance. Like in C++ and other object-oriented programming languages, Python allows for multilevel inheritance implemented at any depth.

Hierarchical Inheritance

A hierarchical inheritance occurs when quite one class springs from one parent or base class. The features inherited by the sub-class or the kid class are included within the parent class. What sets hierarchical inheritance aside from multi-level inheritance is that the order during which the connection between the classes is established. In multilevel inheritance, the order are often haphazard and parent classes can inherit features from previous child classes.

Why is Inheritance Useful in Python Programming?

Inheritance may be a very handy feature of object-oriented programming because it allows a programmer to simply adhere to at least one of software development's most important– Don't Repeat Yourself (DRY). Simply put, implementing class inheritance in your programs is that the most effective thanks to get more through with fewer lines of code and fewer repetition.

Inheritance also will compel you to pay closer attention to the planning phase of programming to make sure that you simply write a program code that's clear, minimalist, and effective.

Another use of inheritance is adding functionality to varied sections of your program.

Inheritance Example

In Python, this is often done by deriving classes. Let's say we've class called SportsCar.

```
class Vehicle(object):

def __init__(self, makeAndModel, prodYear, airConditioning):

self.makeAndModel = makeAndModel

self.prodYear = prodYear

self.airConditioning = airConditioning

self.doors = 4

def honk(self):

print "%s says: Honk! Honk!" % self.makeAndModel
```

Now, below that, create a replacement class called SportsCar, but rather than deriving

objec t , we're getting to derive from Vehicle.

```
class SportsCar(Vehicle)

def__init__(self, makeAndModel, prodYear, airConditioning):

self.makeAndModel = makeAndModel

self.prodYear = prodYear
```

self.airConditioning = airConditioning

self.doors = 4

Leave out the honk function, we only need the constructor function here.

Now declare a sports car. I'm just getting to accompany the Ferrari.

ferrari = SportsCar("Ferrari Laferrari", 2016, True)

Now test this by calling

ferrari.honk()

and then saving and running. It should explode without a hitch. Why is this? this is often because the notion of inheritance says that a toddler class derives functions and sophistication variables from a parent class. Easy enough concept to understand . subsequent one may be a little tougher.

Class Polymorphism and Abstraction

In computing , polymorphism and abstraction are advanced programming features that reach the appliance and usefulness of inheritance.

Polymorphism means should a category Y inherit from class X, it doesn't necessarily need to inherit everything from that class. It can implement a number of the inherited methods and attributes differently. Python, being implicitly polymorphic, can overload operators to grant them the specified behavior supported individual contexts. the thought of polymorphism is that an equivalent process are often performed in several ways depending upon the requirements of things . this will be wiped out two alternative ways in Python: method overloading and method overriding .

Method overloading is defining an equivalent function twice with different arguments. for instance , we could give two different initializer functions to our Vehicle class. Right now, it just assumes a vehicle has 4 doors. If we wanted to specifically say what percentage doors a car had, we could make a replacement initializer function

below our current one with another doors argument, like so (the newer one is on the bottom):

def __init__(self, makeAndModel, prodYear, airConditioning):

self.makeAndModel = makeAndModel

self.prodYear = prodYear

self.airConditioning = airConditioning

self.doors = 4

def __init__(self, makeAndModel, prodYear, airConditioning, doors):

self.makeAndModel = makeAndModel

self.prodYear = prodYear

self.airConditioning = airConditioning

self.doors = doors

Somebody now when creating an instance of the Vehicle class can choose whether or not they define the amount of doors or not. If they don't, the amount of doors is assumed to be 4. Method overriding is when a toddler class overrides a parent class's function with its code. for instance , create another class which extends Vehicle called Moped. Set the doors to 0, because that's absurd, and set air con to false. the sole relevant arguments are make/model and production year. It should appear as if this:

class Moped(Vehicle):

def __init__(self, makeAndModel, prodYear):

self.makeAndModel = makeAndModel

self.prodYear = prodYear

self.airConditioning = False

self.doors = 0

Now, if we made an instance of the Moped class and called the honk() method, it might honk. But it's public knowledge that mopeds don't honk, they beep. So let's override the parent class's honk

method with our own. this is often super simple. We just redefine the function within the child class:

def honk(self):

print “%s says: Beep! Beep!” % self.makeAndModel

I’m a part of the 299,000,000 Americans who couldn’t name a make and model of moped if their life trusted it, but you'll test out if this works for yourself but declaring an instance of the Moped class and trying it out.”

Abstraction

Abstraction is usually a net positive for an outsized number of applications that are being written today, and there’s a reason Python and other object-oriented programming languages are incredibly popular. Abstraction is that the process of simplifying complex realities by modeling classes to handle specific problems. An abstract class can't be instantiated and you'll neither create class instances nor objects for them. Abstract classes are designed to inherit all or only specific features from a base class. Abstraction innately makes the language easier to know , read, and learn. Though it makes the language a tad bit less powerful by removing a number of the facility that the user has over the whole computer architecture, this is often traded instead for the power to program quickly and efficiently within the language, not dalliance handling trivialities like memory addresses or things of the likes of . These apply in Python because, well, it’s incredibly simple. You can’t get down into the nitty-gritty of the pc , or do much with memory allocation or maybe specifically allocate an array size too easily, but this is often a tradeoff for amazing readability, a highly secure language during a highly secure environment, and simple use with programming. Compare the following:

snippet of code from C:

#include

int main(void) {

printf(“hello world”);

```
return 0;
}
```

to the Python code for doing the same:

```
print (“hello world”)
# That’s it. That’s all there's thereto .
```

Encapsulation

The last major concept in object-oriented programming is that of encapsulation. This one’s the simplest to elucidate . this is often the notion that common data should be put together, which code should be modular. I’m not getting to spend long explaining this because it’s an excellent simple concept. the whole notion of classes is as concise of an example as you'll get for encapsulation: common traits and methods are bonded together under one cohesive structure, making it super easy to make things of the type without having to make plenty of super-specific variables for each instance. Well, there we go.

Conclusion

Programming is not just about getting a PC to urge things done. it's tied in with composing code that's helpful to people. Great programming is saddling complexity by composing code that rhymes with our instincts. Great code are going to be code that we will use with a negligible amount of setting.

The most important thing for you to try to is to practice programming in Python. If you've got read until here then you've got already absorbed quite much. you would like to practice all the items you've got learned to form sure you consolidate that knowledge (i.e. make it stick).

Knowledge is useless without application. Learning the way to program without actually programming will only waste the time you invested here. it's like learning the way to ride a motorcycle by reading books or articles about it - which will never be enough! you would like to ride a motorcycle to find out the way to ride a motorcycle .

Also, confirm to familiarize yourself with useful resources you'll easily ask once you need help. There are three obvious ones: Python’s documentation , Stack Exchange .

During your programming journey, you'll encounter seemingly impossible problems. Always get help if you encounter those problems. and through those times, never hesitate to succeed in out for help.